R3. Considere uma conexão TCP entre o hospedeiro A e o hospedeiro B. Suponha que os segmentos TCP que trafegam do hospedeiro A para o hospedeiro B tenham número de porta da origem x e número de porta do destino y. Quais são os números de porta da origem e do destino para os segmentos que trafegam do hospedeiro B para o hospedeiro A?
O endereço IP do host de destino e o número da porta do socket de destino.

R4. Descreva por que um desenvolvedor de aplicação pode escolher rodar uma aplicação sobre UDP em vez de sobre TCP.
UDP. Por ele ser mais simples que o TCP, não ser orientado a conexão ele acaba sendo mais rápido que o TCP.

R5. Por que o tráfego de voz e de vídeo é frequentemente enviado por meio do UDP e não do TCP na Internet de hoje? (Dica: A resposta que procuramos não tem nenhuma relação com o mecanismo de controle de congestionamento no TCP.)
Como a maioria dos firewalls são configurados para bloquear o tráfego TCP, o uso do UDP para o tráfego de voz e vídeo permite que o tráfego atravesse os firewalls.

R6. É possível que uma aplicação desfrute de transferência confiável de dados mesmo quando roda sobre UDP? Caso a resposta seja afirmativa, como isso acontece?
Sim. O desenvolvedor da aplicação pode inserir transferência confiável de dados no protocolo da camada de aplicação; entretanto, isto vai exigir uma quantidade significativa de trabalho e testes.

R7. Suponha que um processo no hospedeiro C possua um socket UDP com número de porta 6789 e que o hospedeiro A e o hospedeiro B, individualmente, enviem um segmento UDP ao hospedeiro C com número de porta de destino 6789. Os dois segmentos serão encaminhados para o mesmo socket no hospedeiro C? Se sim, como o processo no hospedeiro C saberá que os dois segmentos vieram de dois hospedeiros diferentes?
Sim, ambos os segmentos serão direcionados para o mesmo socket. Entretanto, para cada segmento recebido, na interface socket, o Sistema Operacional irá fornecer ao processo o endereço IP para determinar a origem dos segmentos individuais.

R8. Suponha que um servidor da Web seja executado no computador C na porta 80. Esse servidor utiliza conexões contínuas e, no momento, está recebendo solicitações de dois computadores diferentes, A e B. Todas as solicitações estão sendo enviadas por meio do mesmo socket no computador C? Se estão passando por diferentes sockets, dois deles possuem porta 80? Discuta e explique.
Para cada conexão persistente, o servidor Web cria sockets separados. Cada socket é identificado com uma tupla quádrupla: (endereço IP de origem, número da porta de origem, endereço IP de destino, número da porta de destino). Quando o host C recebe um datagrama IP ele examina estes quatro campos no datagrama/segmento para determinar a qual socket ele deve entregar os dados do segmento TCP. Assim, as requisições de A e B passam através de sockets diferentes. O identificador para ambos os sockets tem a porta 80 como destino, no entanto, os identificadores para estes sockets têm diferentes valores para o IP de origem. Ao contrário do UDP, quando a camada de transporte passa a carga de um segmento TCP para o processo da aplicação, ele não

especifica o endereço IP de origem, já que isto está implicitamente especificado pelo identificador do socket.

R9. Em nossos protocolos rdt, por que precisamos introduzir números de sequência?
Os números de sequência são necessários para que o receptor possa saber se um pacote recebido contém novos dados ou é uma retransmissão.

R10. Em nossos protocolos rdt, por que precisamos introduzir temporizadores?
Para lidar com as perdas no canal. Se o ACK para um pacote transmitido não é recebido dentro do período de duração do temporizador para o pacote, o pacote (ou a sua ACK ou NACK) é assumido como tendo sido perdido. Assim, o pacote é retransmitido.

R11. Suponha que o atraso de viagem de ida e volta entre o emissor e o receptor seja constante e conhecido para o emissor. Ainda seria necessário um temporizador no protocolo rdt 3.0, supondo que os pacotes podem ser perdidos? Explique.
Um temporizador ainda seria necessário no protocolo rdt 3.0. Se o tempo de ida e volta (RTT) é conhecido então a única vantagem será que o emissor sabe com certeza que o pacote ou o ACK (ou NACK) para o pacote foi perdido em comparação com a situação real, onde esta informação pode estar a caminho para o remetente, após a expiração do timer. No entanto, para detectar a perda para cada pacote, um temporizador de duração constante